Problem 4 (6 Points)

In this problem you will code a function to perform feature filtering using the Pearson's Correlation Coefficient method.

To start, run the following cell to load the mtcars dataset. Feature names are stored in `feature_names`, while the data is in `data`.

In [1]:
```
import numpy as np

feature_names = ["mpg","cyl","disp","hp","drat","wt","qsec","vs","am","gear","carb"]
data = np.array([[21,6,160,110,3.9,2.62,16.46,0,1,4,4], [21,6,160,110,3.9,2.875,17.02,
                [18.1,6,225,105,2.76,3.46,20.22,1,0,3,1], [14.3,8,360,245,3.21,3.57,15
                [17.8,6,167.6,123,3.92,3.44,18.9,1,0,4,4],[16.4,8,275.8,180,3.07,4.07,
                [10.4,8,460,215,3,5.424,17.82,0,0,3,4],[14.7,8,440,230,3.23,5.345,17.4
                [21.5,4,120.1,97,3.7,2.465,20.01,1,0,3,1],[15.5,8,318,150,2.76,3.52,16
                [27.3,4,79,66,4.08,1.935,18.9,1,1,4,1],[26,4,120.3,91,4.43,2.14,16.7,0
                [15,8,301,335,3.54,3.57,14.6,0,1,5,8],[21.4,4,121,109,4.11,2.78,18.6,1
```

Filtering

Now define a function `find_redundant_features(data, target_index, threshold)`.
Inputs:

- data: input feature matrix
- target_index: index of column in `data` to treat as the target feature
- threshold: eliminate indices with pearson correlation coefficients greater than `threshold`

Return:

- Array of the indices of features to remove.

Procedure:

1. Compute correlation coefficients with `np.corrcoeff(data.T)`, and take the absolute value.
2. Find off-diagonal entries greater than `threshold` which are not in the target_index row/column.
3. For each of these entries above `threshold`, determine which has a lower correlation with the target feature -- add this index to the list of indices to filter out/remove.
4. Remove possible duplicate entries in the list of indices to remove.

In [23]:
```
def find_redundant_features(data, target_index, threshold):
    # YOUR CODE GOES HERE
    cc = np.abs(np.corrcoef(data.T))

    remove_features = []
    for i in range(0,len(cc)):
        for j in range(i+1,len(cc)):
                if(cc[i,j] > threshold and j != target_index and i!=target_index):
                    if(cc[target_index,i] < cc[target_index,j]):
                        remove_features.append(i)
                    else:
                        remove_features.append(j)

    remove_features = list(set(remove_features))

    return remove_features
```

Testing your function

The following test cases should give the following results: | target_index | threshold | | Indices to remove | |---|---|---|---| | 0 | 0.9 | | [2] | | 2 | 0.7 | | [0, 3, 4, 5, 6, 7, 8, 9, 10] | | 10 | 0.8 | | [1, 2, 5] |

Try these out in the cell below and print the indices you get.

In [24]:

```
# YOUR CODE GOES HERE

features_0 = find_redundant_features(data,0,0.9)
features_2 = find_redundant_features(data,2,0.7)
features_10 = find_redundant_features(data,10,0.8)


print("for target index = 0 the indices to remove are: ",features_0)
print("for target index = 2 the indices to remove are: ",features_2)
print("for target index = 10 the indices to remove are: ",features_10)
```

```
for target index = 0 the indices to remove are:  [2]
for target index = 2 the indices to remove are:  [0, 3, 4, 5, 6, 7, 8, 9, 10]
for target index = 10 the indices to remove are:  [1, 2, 5]
```

Using your function

Run these additional cases and print the results: | target_index | threshold | | Indices to remove | |---|---|---|---| | 4 | 0.9 | | ? | | 5 | 0.8 | | ? | | 6 | 0.95 | | ? |

In [25]:

```
# YOUR CODE GOES HERE
features_4 = find_redundant_features(data,4,0.9)
features_5 = find_redundant_features(data,5,0.8)
features_6 = find_redundant_features(data,6,0.95)


print("for target index = 4 the indices to remove are: ",features_4)
print("for target index = 5 the indices to remove are: ",features_5)
print("for target index = 6 the indices to remove are: ",features_6)
```

```
for target index = 4 the indices to remove are:  [1]
for target index = 5 the indices to remove are:  [0, 1, 3, 7]
for target index = 6 the indices to remove are:  []
```

In [ ]: